

令和元年（ワ）第10940号 損害賠償請求事件

原告 森次 茂廣

被告

準備書面 19

令和4年10月13日

大阪地方裁判所第26民事部合議係 御中

被告訴訟代理人弁護士

第1 騒音計の出力電圧と、乙23のソースコード

- 1 先日の期日において、専門委員から、被告の第18準備書面別紙12頁記載の「表.1 NL-21 出力電圧とデジタル値の関係」のデジタル値を、乙23の28頁目以下、原告第11準備書面3頁目の図1のコードの中に記載された計算に当てはめても、正しい値が出ない、との指摘があった。

正しい値が出ない理由は、NL-21の出力電圧及びデジタル値を、乙23のソースコードに当てはめているためである。

乙23のソースコードの騒音計は、NL-14であって、NL-21ではない。

乙23のソースコード13頁、16頁に「1CH 騒音計レンジ(NL-14)」を使用するコードであることが記載されている。

そして、NL-14 の仕様と、NL-21 の仕様は異なる。

NL-14 の測定レンジは「積分形精密騒音計 NL-14 取扱説明書」の仕様より、次頁の図. 1 の赤枠で示すように 20～80dB、30～90dB、40～100dB、50～110dB、60～120dB、70～130dB、80～140dB の 7 段階から選択できるとなっている。

ちなみに、NL-21 の測定レンジは「NL-21 取扱説明書」の仕様より、20～80dB、20～90dB、20～100dB、20～110dB、30～120dB、40～130dB の 6 段階から選択する。

仕様	
レベルレンジ	10 dB ステップ、7段 20～80 dB、30～90 dB、40～100 dB、50～110 dB、 60～120 dB、70～130 dB、80～140 dB フィルターユニット使用時は10～70 dBの設定が可能
実効値指示特性	波高率3：誤差0.5 dB以下(フルスケール) 波高率10：誤差1.5 dB以下(フルスケール-10 dB)
演算	デジタル方式 サンプリング周期 10 ms (L_{eq} 、 L_E 、 L_{max})、100 ms (L_x)
交流出力端子	出力電圧：1 V rms(フルスケールにおいて) 出力抵抗：約600 Ω 負荷抵抗：10 kΩ以上
直流出力端子	出力電圧：3.5 V(フルスケールにおいて)、0.5 V/10 dB 出力抵抗：約50 Ω 負荷抵抗：10 kΩ以上
I/O 端子	機能：コンピューターによる騒音計の制御とデータ出力 プリンター CP-10 へのデータ出力 レベルレコーダー LR-06 への測定条件の出力 レベルレコーダー LR-04/LR-06 によるフィルターの制御 RS-232-C インタフェース 伝送方式： 伝送制御手順 有手順 通信方式 調歩同期、半二重 データビット 8ビット ストップビット 2ビット パリティ なし 通信速度 4800 bps

図.1 測定レンジ・出力電圧

そして、出力電圧については「積分形精密騒音計 NL-14 取扱説明書」

の仕様より、上記図.1 の矢印で示すように「3.5V (フルスケール)、

0.5V/10dB」で出力されると記載されている。

つまり、NL-14 では、測定レンジの最大値が測定されると 3.5V の電圧が出力され、測定値が 10(dB)変化すると出力電圧は 0.5V 変化するのであり、7段階ある測定レンジのうち、現場の状況にあった最適な測定レンジを選択してサイレントロボを使用する。

この選択された測定レンジでの測定値と出力電圧の関係を図示すると次の図.2～図.8 のようになる。

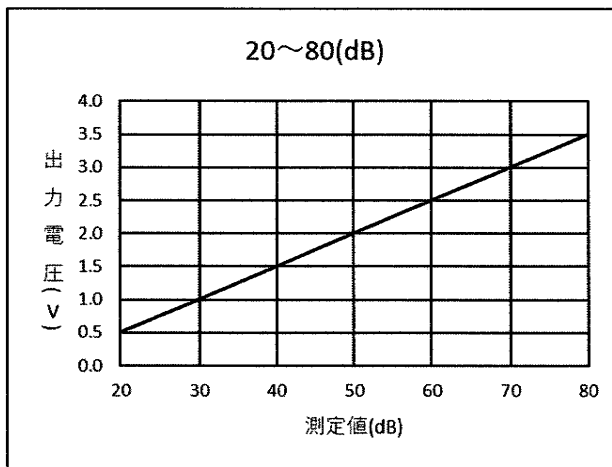


図.2

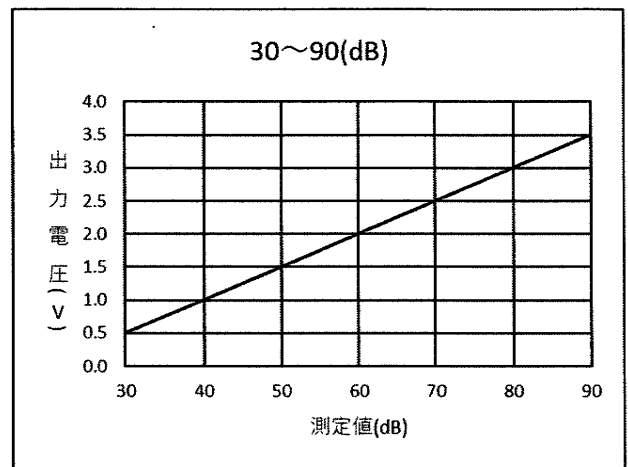


図.3

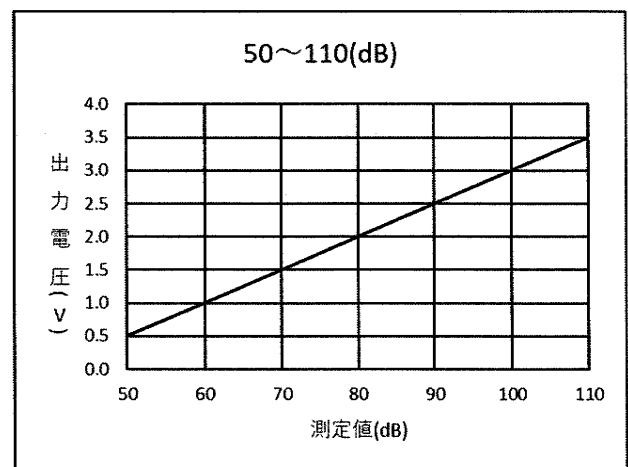
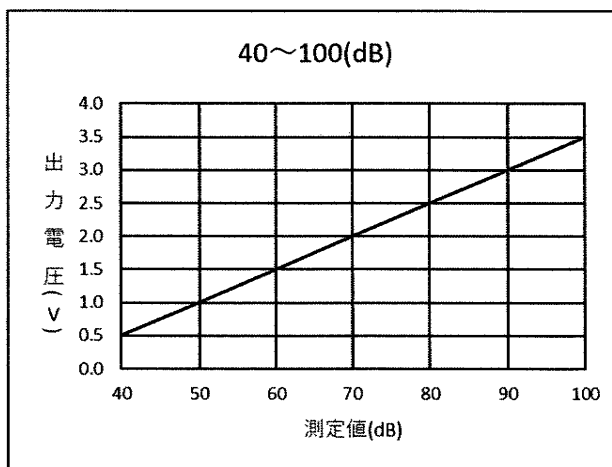


図.4

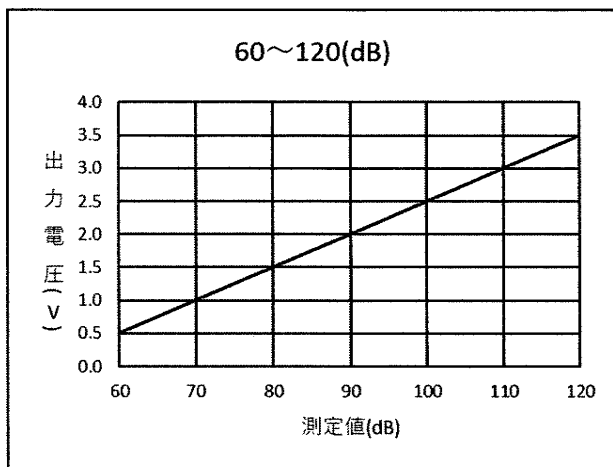


図.5

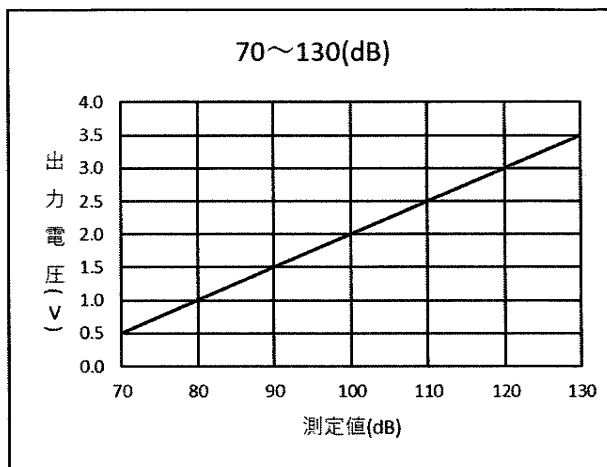


図.6

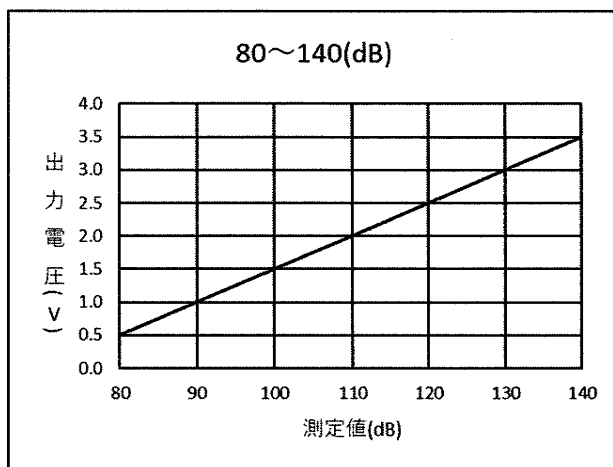


図.7

図.8

以上のとおり、NL-14 と NL-21 とは、測定レンジが異なるのであり、その結果、それに応じた測定値と出力電圧の値も異なる。

乙23は、NL-14 の出力電圧値を前提としたソースコードであり、異なる出力電圧値である NL-21 では乙23のソースコードは正常に作動しない。

なお、被告が第18準備書面別紙12頁にてNL-21の表を示したのは、裁判所の求釈明事項(2)がNL-21のデジタル変換を乙31に基づき説明することを求められたことにあり、これは、被告から当該求釈明事項(2)のNL-21はNL-14ではないかとの確認をすべきであった。

第2 乙23号証以降のソースコードの変遷

乙23号証以降のソースコードの変遷は次の通りである。

乙23号証以降のソースコードの変遷

年月日	サイレントロボ構成機器・プログラム			特記事項	備考
	振動計	騒音計	プログラム		
平成 16 年 2 月 6 日	VM-52	NL-14	H16.2.6 版	・乙 23 号証ソースコード	
平成 16 年 7 月 1 日	VM-52	NL-21	H16.7.1 版	・騒音計レンジ変更	
平成 16 年 8 月 2 日	VM-53A	NL-21	H16.8.2 版	・振動計レンジ変更	
平成 16 年 8 月 11 日	VM-53A	NL-21	H16.8.11 版	・瞬時値、L05、L10 への切り替え機能追加	
平成 16 年 9 月 22 日	VM-53A	NL-21	H16.9.22 版	・外部表示器への電光表示	乙 4 号証
平成 17 年 2 月 6 日	VM-53A	NL-21	H17.2.6 版	・Leq の統計処理機能追加	

※プログラムの版の記載の日付は、exe ファイルの日付。

第3 サンプルングピッチを90msに設定している点について

- 1 騒音測定や振動測定の方法はJISで規定されているものの、サンプルングピッチやデータ個数については明記されていない。

そこで、当社では、サイレントロボのプログラム開発に当たり、従来の時間率騒音・振動レベルのデータ取得・処理方法に対し、1秒間に10個のデータを取得し、その最大値（乙23号証ソースコードではバグで最小値となっている）を求めて1秒に1個の騒音・振動レベルとして取り扱い、その騒音・振動レベルの過去10分間（評価時間）の600データに対し、0.1dbスライスで累積度数を求めている。

そして、時間率騒音レベルL05は騒音レベルの累積度数から90%レンジ上端値を求め、時間率振動レベルL10は振動レベルの累積度数から80%レンジ上端値として求めて、出力表示している。

- 2 ところが、サンプルングピッチを100msに設定してテストさせると1秒間に10個のデータが取得できなかった。

その原因は、CPUの処理速度によるものと思われたところ、被告は、1秒間に10個のデータを採ること自体に重点を置くことに

し、また10個のデータの最後の間隔が均等でなくても処理結果は微差と評価できるとして、確実に1秒間に10個のデータを採れる90msに設定した。

3 処理結果が微差と評価できる理由は、時間率騒音・振動レベル(L05、L10)の処理方法による。

- (1) 時間率騒音・振動レベル(L05、L10)の処理方法について、電算処理が行われていなかった時代は、チャート紙に記録された騒音・振動レベル波形線が、一定間隔(例:1s、5s)の時間軸と交わる点(交点)の騒音・振動レベルの値を、50個~100個読み取り、その交点の騒音・振動レベルの値を計算処理して、時間率騒音・振動レベル(L05、L10)を求めていた。
- (2) 例えば、図1のような騒音レベルのデータが記録されたとする。

この図1の騒音レベル波形から、ある一定間隔（図1の横軸の間隔が、一定間隔を示す。）の波形線の交点（図1の、波形線と縦線が交わる点）の騒音レベル値を読み取る。

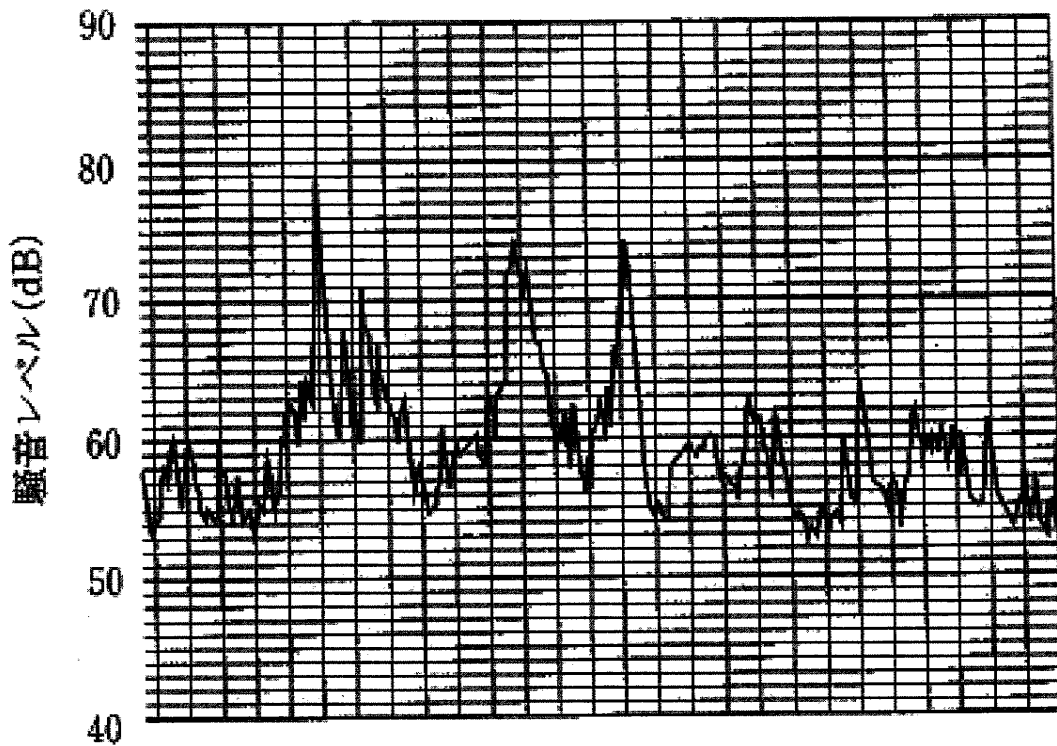
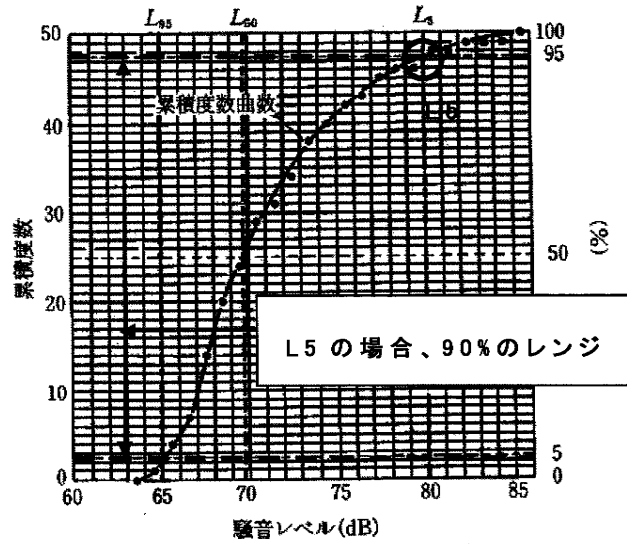


図1 騒音レベル波形

次に、読み取ったデータを測定用紙（図2a）に記録するとともに1dB毎の累積度数を求め、累積度数曲線（図2b）を作図する。その90%レンジ上端値（L05）を読み取ることで、時間率騒音・振動レベル（L05、L10）を処理していた。

測定の対象、場所、条件など										91年8月1日(月) 2:30 A.M. (2)	
a点、環境騒音										天気晴、微風	
										測定器 a社 b型	
										測定者 T.H	
										調整修正 (A) B, C	
	1	2	3	4	5	6	7	8	9	10	
(A)	71	72	64	65	67	66	67	68	70	73	
	73	70	78	69	68	67	67	72	74	80	
	76	77	66	85	65	67	68	73	69	70	
	71	72	70	67	75	67	68	65	80	77	
	74	73	70	68	82	75	66	67	68	69	
	測定回数	0	1	2	3	4	5	6	7	8	9
(B)	60音					0	1	4	7	14	20
	70音	5	2	3	4	2	2	1	2	1	0
	80音	2	0	1	0	0	1				
	90音										
(C)	騒音レベル、中央値(90%レンジ) 70 (65,80) dB										

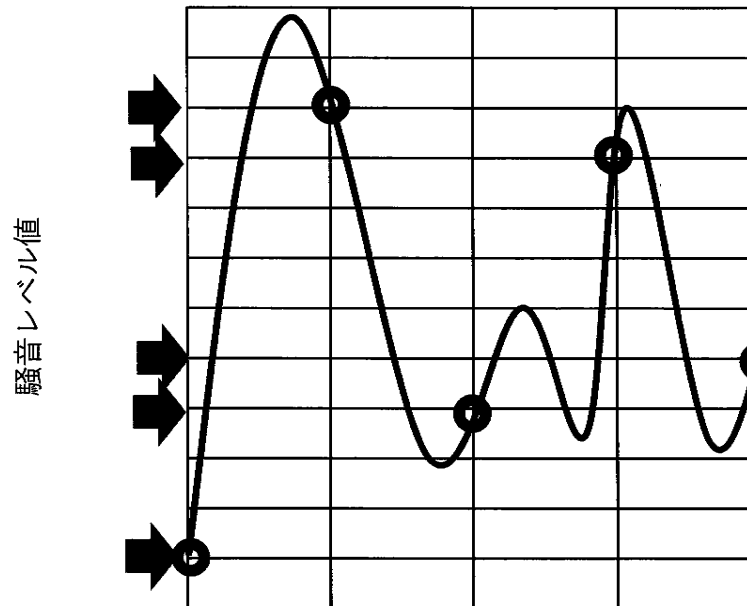


(a) 測定用紙例

(b) 累積度数曲線(例)

図2 時間率騒音レベルの求め方

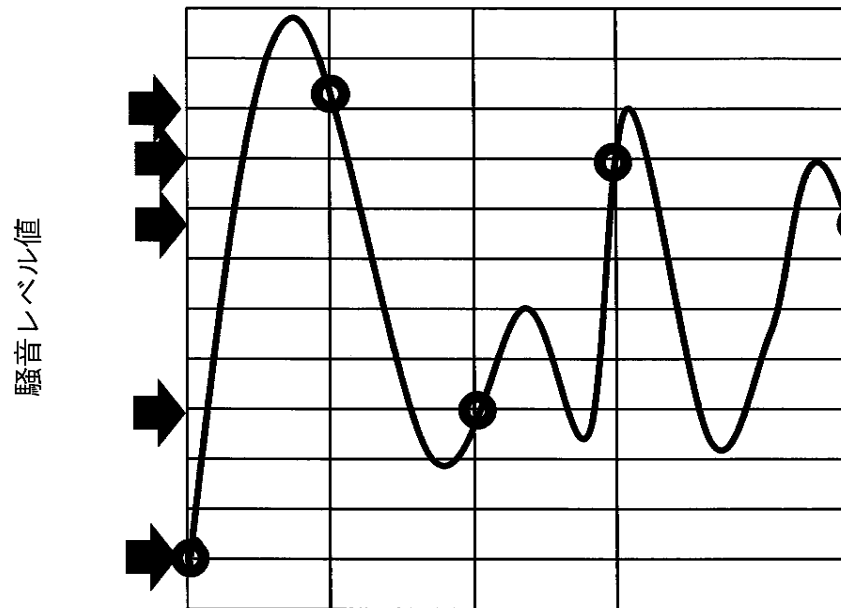
- (2) つまり、騒音レベル値を読み取る位置は、交点であって、波形の線上全てでも、波形の頂点でもない。



二重丸が交点。

矢印は、交点のある騒音レベル値。

そのため、サンプリングピッチを 90 ms にした場合、 1 秒 を 10 分割した際の最後の間隔が他の間隔と比べて長くなるが、それは、最後の採る交点の位置が一つ異なることを意味する。



二重丸が交点。

矢印は、交点のある騒音レベル値。

もっとも、その1個だけ異なった交点の騒音レベル値が、1秒間のうち最も高い又は低い騒音レベル値とは限らない。また、仮に、その一個異なった交点の騒音レベル値が1秒間のうち最も高い又は低い騒音レベル値に当たったとしても、その後に、10分（600秒）分の累積度数を求めて累積度数曲線（図2b）を作図し90%レンジ上端値（L05）を読み取る一連の処理を行うのであり、1個の交点の相違が、全体に与える影響は微

差と評価できる。

第4 佐伯専門員の質問に対する回答

- 1 佐伯専門員から、「乙23号証28ページ7行目～16行目のソースコード、特に、8行目、10行目、13行目及び15行目の、各「WData」で始まるコードに関して、InpBuf()のデータは、Integerで宣言（乙23号証17ページ7行参照。）されており、整数値を有するものである。そして、上記コードでは、「 $(\text{InpBuf}(ii*2) - 32768)$ 」という整数値を「3276.8」で除しているが、かかる計算結果が実数値で処理されているのか、整数値で処理されているのかが不明である。Visual Basic6.0のデフォルト処理として、このような場合にどのようなデータ型で処理が行われ、WDataの各計算段階において、適切なデータ型の取り扱いとなっているのか、かつ、求めるべき値を精度よく算出するものであるのか、順を追って説明してください。」との質問を受けた。

これに対し、次のとおり回答する。

2 暗黙の型変換

Visual Basic 6.0は異なる数値型の計算式を実行する時、自動

的に、表現できる数値の範囲が広いほうの型に他の型が変換される

(これを「暗黙の型変換」と言う)。

例えば以下のようなプログラムコードがあるとする。

```
Dim Ver1 As Integer
```

```
Dim Ver2 As Single
```

```
Ver1 = 5
```

```
Ver2 = Ver1 * 2.5
```

変数「Ver2」は変数「Ver1」×2.5 の計算結果を保存している。

この時、Ver1 は「整数型」で宣言されているが「2.5」は小数部があるため「単精度浮動小数点型」の定数として扱われる。そのため上記「暗黙の型変換」が行われ、表現できる数値の範囲が狭い「整数型」から表現できる数値の範囲が広い「単精度浮動小数点型」に変数「Ver1」の数値を変換し計算が行われる。結果として「単精度浮動小数点型」変数「Ver2」には「単精度浮動小数点型」で計算された結果「12.5」が保存される。

なお、主な数値型と表現できる数値の範囲を以下に示す。

整数型 $-32768 \sim 32767$

長整数型 $-2147483648 \sim 2147483647$

単精度浮動小数点型 $-3.402823E+38 \sim -1.401298E-45$

(負の値)

$1.401298E-45 \sim 3.402823E+38$

(正の値)

倍精度浮動小数点型 $-1.79769313486232E+308$

$\sim -4.94065645841247E-324$

(負の値)

$4.94065645841247E-324$

$\sim 1.79769313486232E+308$

(正の値)

3 乙第 23 号証 28 ページ 8 行目の計算過程の説明

次に、乙第 23 号証 28 ページ 8 行目の計算過程を説明する。

wData(ii, 0)

$$= \text{Int}(((\text{InpBuf}(ii * 2) - 32768) / 3276.8 * 20 + (\text{RangeSouonnDb} - 70)) * 10) / 10$$

以上の式を、Step1～Step5 に分解して説明する。

(1) 【Step1】 InpBuf(ii * 2) - 32768 の計算

InpBuf() は「整数型」で宣言されているが、定数「32768」が「整数型」では表現範囲を超えるため、「長整数型」定数として扱われ、暗黙の型変換により InpBuf() が「長整数型」に変換され、計算結果は「長整数型」となる。

(2) 【Step2】 「Step1 の結果」 / 3276.8 * 20 の計算

「Step1 の結果」は、「長整数型」、定数「3276.8」は「単精度浮動小数点型」、定数「20」は「整数型」となるが、暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

これは「長整数型」の数値を「単精度浮動小数点型」に変換した場合、有効桁数が不足し情報が欠落する場合があるためである。

- (3) 【Step 3】 「Step2 の結果」 + (RangeSouonnDb - 70)
の計算

「Step2 の結果」は「倍精度浮動小数点型」RangeSouonnDb
は「単精度浮動小数点型」、定数「70」は「整数型」となるが暗
黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は
「倍精度浮動小数点型」となる。

- (4) 【Step 4】 Int(「Step3 の結果」 * 10) の計算

「Step3 の結果」は「倍精度浮動小数点型」、定数「10」は
「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」
に変換され、Int()関数の戻り値は「倍精度浮動小数点型」とな
る。

- (5) 【Step 5】 wData(ii, 0) = 「Step4 の結果」 / 10 の計
算

「Step4 の結果」は「倍精度浮動小数点型」、定数「10」は
「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」
に変換され、計算結果は「倍精度浮動小数点型」となる。

wData()は「単精度浮動小数点型」で宣言されているため、計
算結果を保存する前に「倍精度浮動小数点型」から「単精度浮動
小数点型」に型変換が行われる。

この結果、有効桁数の減少により情報が失われるが本プログラ
ムでは小数点以下1桁の数値しか使用しないため問題になる事は
ない。

4 乙第 23 号証 28 ページ 10 行目の計算過程の説明

$$\begin{aligned} & \text{wData(ii, 0)} \\ &= \text{Int}(((\text{InpBuf(ii * 2) + 32768}) / 3276.8 * 20 + \\ & \quad (\text{RangeSouonnDb} - 70)) * 10) / 10 \end{aligned}$$

以上の式を, Step1~Step5 に分解して説明する。

(1) 【Step 1】 InpBuf(ii * 2) + 32768 の計算

InpBuf()は「整数型」で宣言されているが、定数「32768」が「整数型」では表現範囲を超えるため「長整数型」定数として扱われ、暗黙の型変換により InpBuf()が「長整数型」に変換され、計算結果は「長整数型」となる。

(2) 【Step 2】 「Step1 の結果」 / 3276.8 * 20 の計算

「Step1 の結果」は「長整数型」、定数「3276.8」は「単精度浮動小数点型」、定数「20」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

これは「長整数型」の数値を「単精度浮動小数点型」に変換した場合、有効桁数が不足し情報が欠落する可能性があるためである。

(3) 【Step 3】 「Step2 の結果」 + (RangeSouonnDb - 70) の計算

「Step2 の結果」は「倍精度浮動小数点型」、RangeSouonnDb は「単精度 浮動小数点型」、定数「70」は「整数型」となるが

暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

(4) 【Step 4】 `Int(「Step3 の結果」 * 10)` の計算

「Step3 の結果」は「倍精度浮動小数点型」、定数「10」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、`Int()`関数の戻り値は「倍精度浮動小数点型」となる。

(5) 【Step 5】 `wData(ii, 0) = 「Step4 の結果」 / 10` の計算

「Step4 の結果」は「倍精度浮動小数点型」、定数「10」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

`wData()`は「単精度浮動小数点型」で宣言されているため、計算結果を保存する前に「倍精度浮動小数点型」から「単精度浮動小数点型」に型変換が行われる。

この結果、有効桁数の減少により情報が失われるが本プログラムでは小数点以下1桁の数値しか使用しないため問題になる事はない。

5 乙第 23 号証 28 ページ 13 行目の計算過程の説明

$$\begin{aligned} & \text{wData(ii, 1)} \\ & = \text{Int}(((\text{InpBuf(ii * 2 + 1)} - 32768) / 3276.8 * 20 + \\ & \quad (\text{RangeSindouDb} - 60)) * 10) / 10 \end{aligned}$$
(1) 【Step 1】 $\text{InpBuf(ii * 2 + 1)} - 32768$ の計算

InpBuf() は「整数型」で宣言されているが定数「32768」が「整数型」では表現範囲を超えるため「長整数型」定数として扱われ、暗黙の型変換により InpBuf() が「長整数型」に変換され、計算結果は「長整数型」となる。

(2) 【Step 2】 「Step1 の結果」 / $3276.8 * 20$ の計算

「Step1 の結果」は「長整数型」、定数「3276.8」は「単精度浮動小数点型」、定数「20」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

これは「長整数型」の数値を「単精度浮動小数点型」に変換した場合、有効桁数が不足し情報が欠落する可能性があるためである。

(3) 【Step 3】 「Step2 の結果」 + $(\text{RangeSindouDb} - 60)$ の計算

「Step2 の結果」は「倍精度浮動小数点型」、 RangeSindouDb は「単精度浮動小数点型」、定数「60」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

(4) 【Step 4】 Int(「Step3 の結果」 * 10) の計算

「Step3 の結果」は「倍精度浮動小数点型」、定数「10」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、Int()関数の戻り値は「倍精度浮動小数点型」となる。

(5) 【Step 5】 wData(ii, 1) = 「Step4 の結果」 / 10 の計算

「Step4 の結果」は「倍精度浮動小数点型」、定数「10」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

wData()は「単精度浮動小数点型」で宣言されているため、計算結果を保存する前に「倍精度浮動小数点型」から「単精度浮動小数点型」に型変換が行われる。

この結果、有効桁数の減少により情報が失われるが本プログラムでは小数点以下1桁の数値しか使用しないため問題になる事はない。

6 乙第 23 号証 28 ページ 15 行目の計算過程の説明

$$\begin{aligned} & \text{wData(ii, 1)} \\ &= \text{Int}(((\text{InpBuf(ii * 2 + 1) + 32768}) / 3276.8 * 20 + \\ & \quad (\text{RangeSindouDb} - 60)) * 10) / 10 \end{aligned}$$

以上の式を，Step1～Step5 に分解して説明する。

(1) 【Step 1】 InpBuf(ii * 2 + 1) + 32768 の計算

InpBuf()は「整数型」で宣言されているが定数「32768」が「整数型」では表現範囲を超えるため「長整数型」定数として扱われ、暗黙の型変換により InpBuf()が「長整数型」に変換され、計算結果は「長整数型」となる。

(2) 【Step 2】 「Step1 の結果」 / 3276.8 * 20 の計算

「Step1 の結果」は「長整数型」、定数「3276.8」は「単精度浮動小数点型」、定数「20」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

これは「長整数型」の数値を「単精度浮動小数点型」に変換した場合、有効桁数が不足し情報が欠落する可能性があるためである。

(3) 【Step 3】 「Step2 の結果」 + (RangeSindouDb - 60) の計算

「Step2 の結果」は「倍精度浮動小数点型」、RangeSindouDb は「単精度浮動小数点型」、定数「60」は「整数型」となるが暗黙

の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

(4) 【Step 4】 $\text{Int}(\text{「Step 3 の結果」} * 10)$ の計算

「Step 3 の結果」は「倍精度浮動小数点型」、定数「10」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、 $\text{Int}()$ 関数の戻り値は「倍精度浮動小数点型」となる。

(5) 【Step 5】 $\text{wData}(\text{ii}, 1) = \text{「Step 4 の結果」} / 10$ の計算

「Step 4 の結果」は「倍精度浮動小数点型」、定数「10」は「整数型」となるが暗黙の型変換により「倍精度浮動小数点型」に変換され、計算結果は「倍精度浮動小数点型」となる。

$\text{wData}()$ は「単精度浮動小数点型」で宣言されているため、計算結果を保存する前に「倍精度浮動小数点型」から「単精度浮動小数点型」に型変換が行われる。

この結果、有効桁数の減少により情報が失われるが本プログラムでは小数点以下 1 桁の数値しか使用しないため問題になる事はない。

以 上